Information Retrieval and the Internet

Search Engine: Vanilla System

University of Ottawa

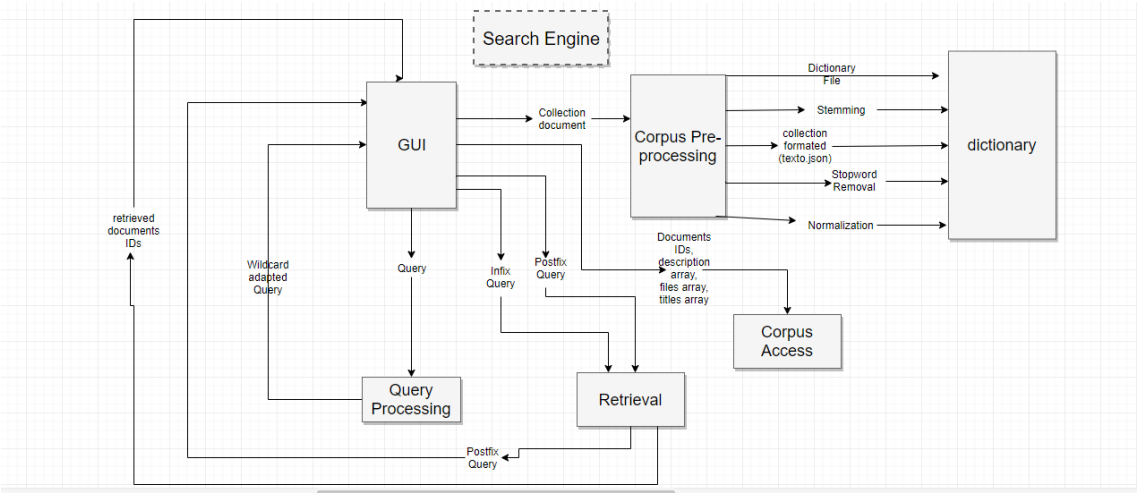
uOttawa

Student: Ilan Daniele

Student Number: 300096342

Index

DIAGRAM OF SYSTEM ARCHITECTURE



MODULES

Corpus Pre-processing

Functionalities/Limitations (cases not handled):

This module receives an html specific file, in this case the courses collection in only one file, and it creates by it another new file in json format, with all the courses formatted to fit one file. The limitation is that it can only work with this specific file.

Problems encountered (if any, as you developed the module):

There was a problem with the encoding, that when it reads from the html, some symbols would be parsed and when it was finally written on the json it gave a word with strange symbols. It could be solved by adding an encoding while opening each file.

User Interface

Functionalities/Limitations:

The query can be typed in and we can choose to use either Boolean Retrieval Model or Vector Space Model. There is the option to press on three collections and look on them, but only uofo courses option work.

By typing the query, and pressing search on the collection, the system pop ups a new window showing the list of documents related. And if you press one of them it will pop up a new window with the specific document data.

Problems encountered:

There was an issue when trying to click on the display of documents to open an specific document. When you searched for press automatically it would generate the list and open each specific document instead of letting you choose which one you would like to open. This was solved.

Dictionary Building

Functionalities/Limitations:

This module receives the specific corpus file that was made before, and by it creates a dictionary variable which in the end will be written on a json file. It would only work for courses corpusFile. There is also an option which allow us to use stemming, normalization or stopword removal in the dictionary, it's called by the corpusPreProcessing module with this parameters, it is only matter of changing for 1 or 0 the values of the call.

It also removes non letter characters like ')', '(', '"', '/', '&', ',', ';', ':'.

In the end it writes the content of the dictionary variable to the file called 'diccionario.json'

Problems encountered:

There where some issues when trying to format the words and delete specific characters, but it was solved.

Inverted Index Construction

Functionalities/Limitations:

This module was actually developed on the dictionary class. In compliment with the dictionary module and the weight calculation of the vector space model, the file called 'diccionario.json' is created which will contain this inverted index.

Problems encountered:

No major issues encountered. Just at the beginning it was a bit confusing where to code and how to organize the functions.

Corpus Access

Functionalities/Limitations:

In this module, given the array of documents i want to retrieve and the corpus file, we would use it to navigate through the file and retrieve the data to the gui.

Problems encountered:

The major issue was how to retrieve the data so it can be accessed in the GUI. So, it was decided to copy each content of each courses in a new file for each, in a new directory, so it can be used in the GUI and the data do not dissappear.

Boolean Retrieval Model

Functionalities/Limitations:

It is implemented in the Retrieval.py file. First, the infix query it's received, by a postfix transformation function, the query is transformed to postfix format so it can be processed easily later. Then, using an evaluation function, the postfix query is passed and by accessing

the dictionary, and then using a Stack and some corresponding operations, the corresponding documents IDs are retrieved to the gui so it can further use it.

Problems encountered:

There was a trouble with the infix to postfix transformations, with specific symbols. The function used which was taken from a site, it was not working properly so it had to be modificated. Same with the postfix evaluation function which also had to be modificated.

Vector Space Model (Weight Calculation)

Functionalities/Limitations:

This module was developed on the dictionary class. Once the dictionary variable was filled, we took out every term of the inverted index and we calculate the term frequency seeing how many times appears, and then we would use the term frequency

Problems encountered:

For this module, the only issue was how and where to calculate the weight, which i actually did it while i was building the dictionary, and the fact that it's made use of several structures to hold the data. No other major issues encountered.

Vector Space Model (Retrieval)

Functionalities/Limitations: as the boolean retrieval model, this module is implemented in the retrieval.py class. It makes use of the already known dictionary to retrieve the specific documents ids and it's weights. Then applying the corresponding vector space model operations the scores of each file are obtained for further use.

In the end an array of documents ordered by most relevant to least relevant is retrieved.

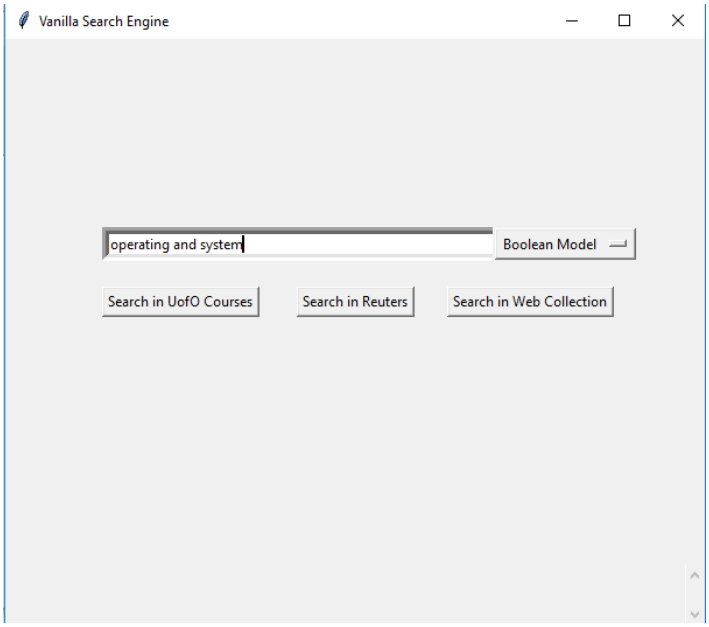
Problems encountered

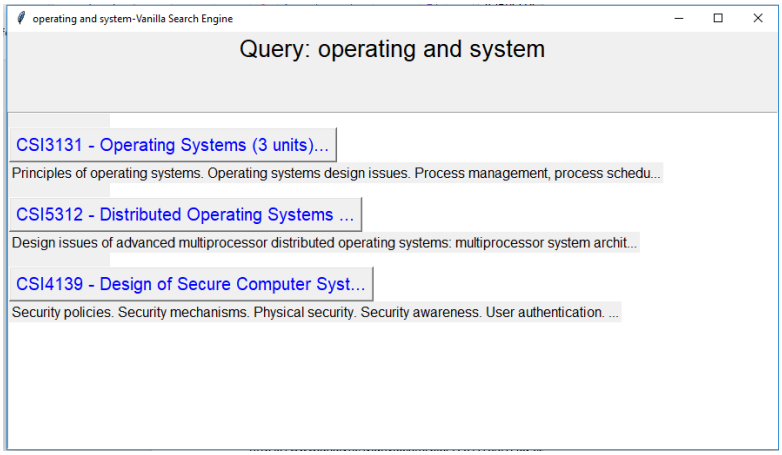
Use of many structures in order to get the weights from the inverted index, do the corresponding operations and then showing them ordered by Rank. The code could be confussing, but works properly.

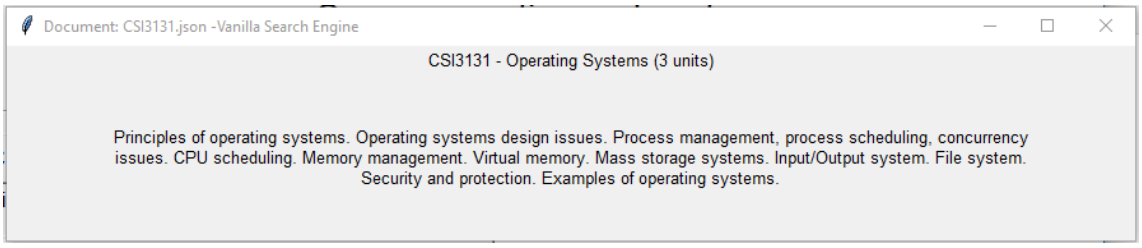
RESULTS

Boolean Retrieval Model:

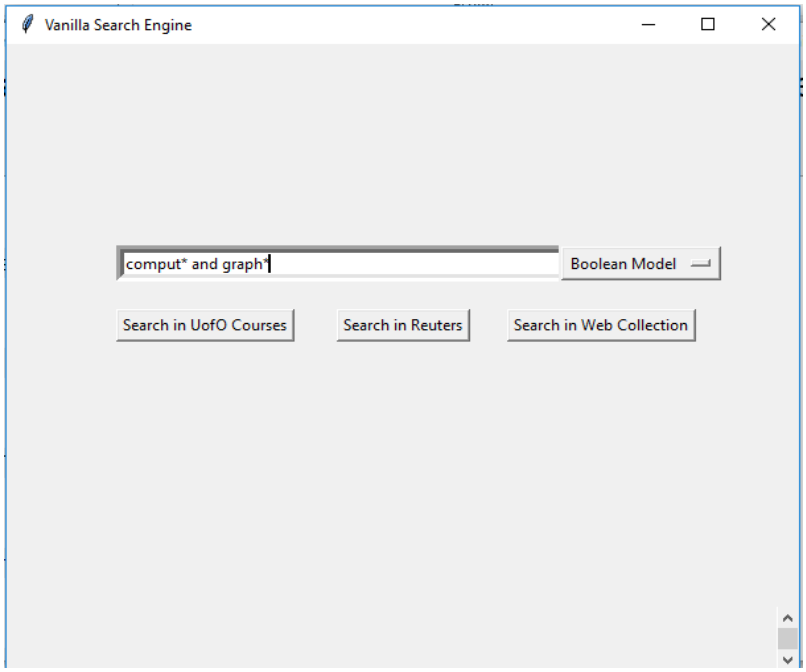
- Query: operating AND system

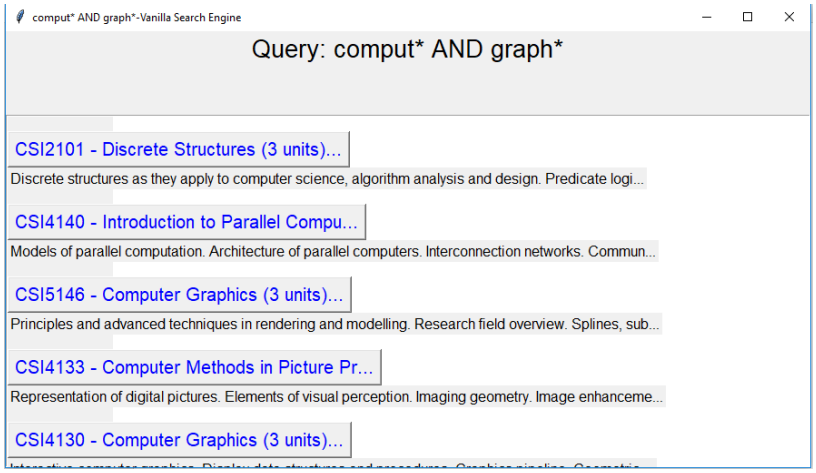




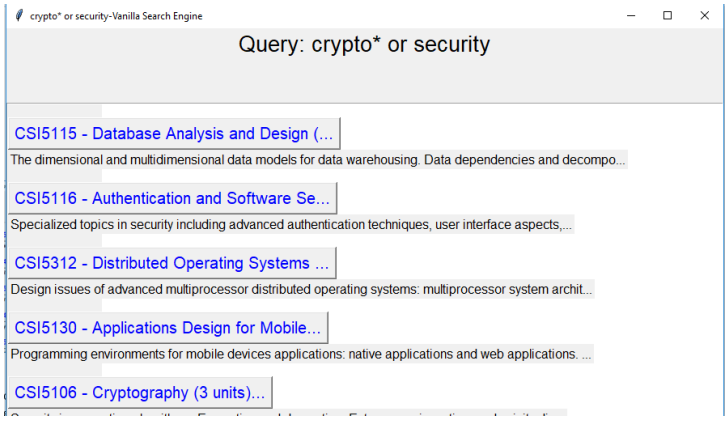


- Query: comput* AND graph*



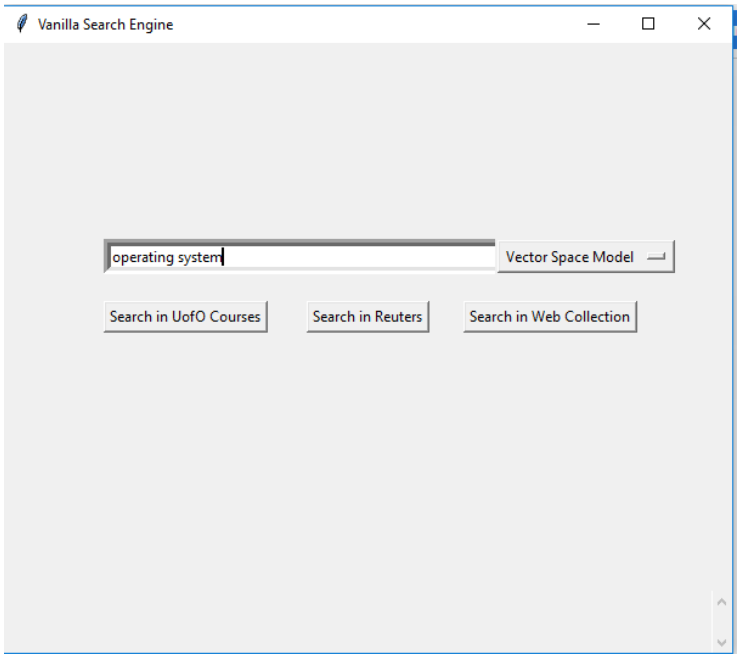


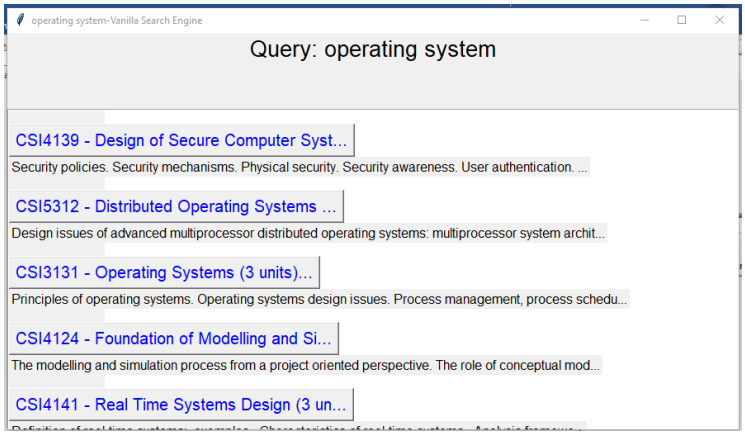
- crypto* OR security

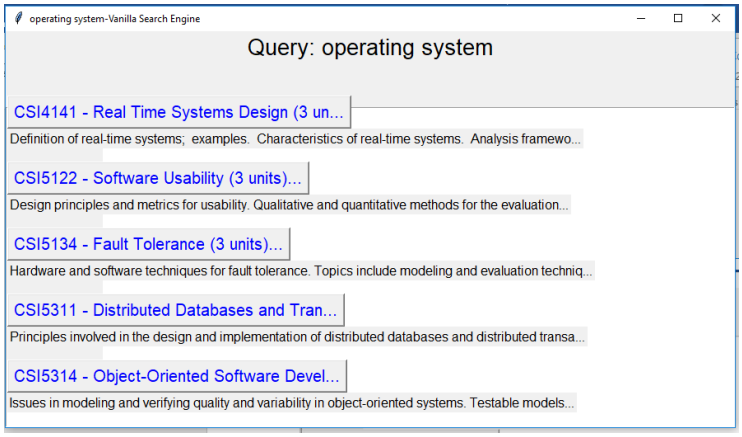


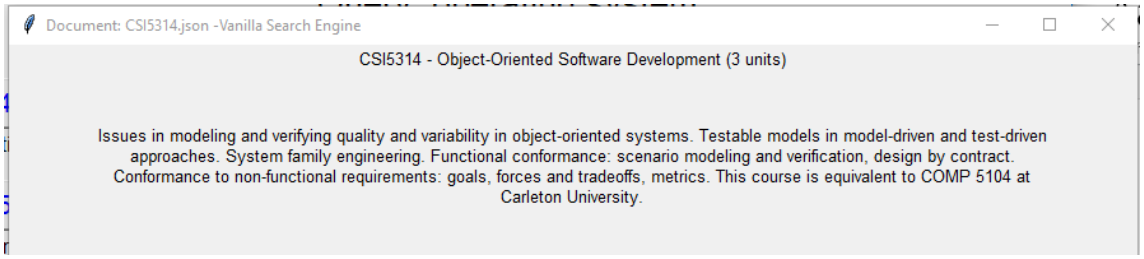
Vector Space Model:

Query: operating system

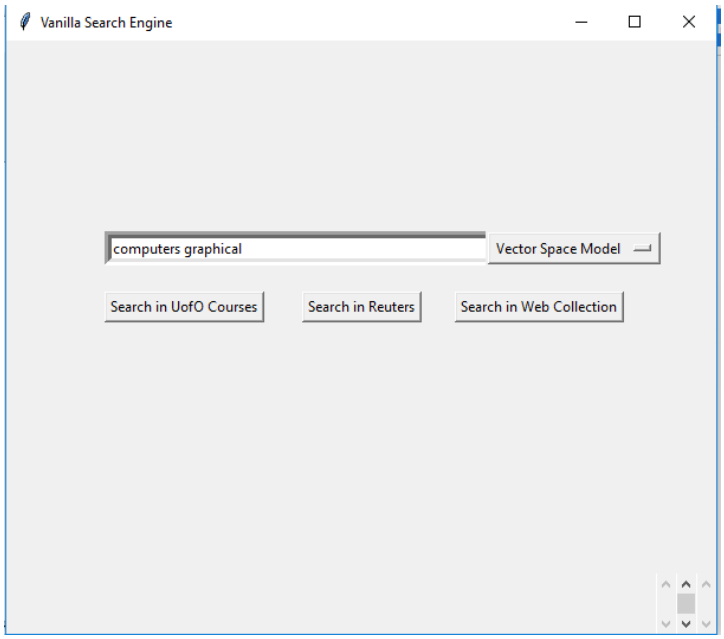

Vanilla Search Engine
operating system
Vector Space Model
Search in UofO Courses
Search in Reuters
Search in Web Collection

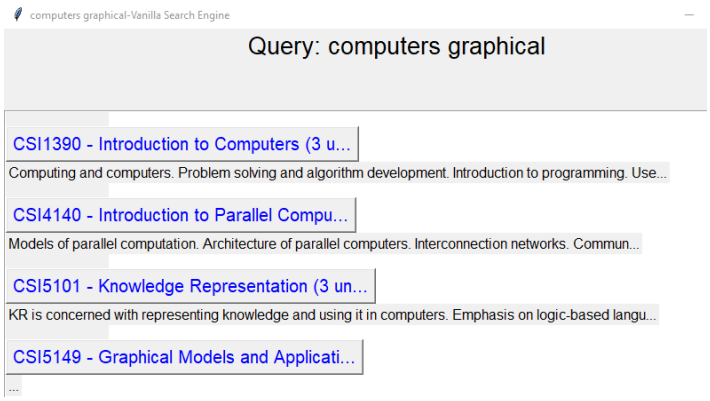

operating system-Vanilla Search Engine
Query: operating system
CSI4139 - Design of Secure Computer Syst...
Security policies. Security mechanisms. Physical security. Security awareness. User authentication. ...
CSI5312 - Distributed Operating Systems ...
Design issues of advanced multiprocessor distributed operating systems: multiprocessor system archit...
CSI3131 - Operating Systems (3 units)...
Principles of operating systems. Operating systems design issues. Process management, process schedu...
CSI4124 - Foundation of Modelling and Si...
The modelling and simulation process from a project oriented perspective. The role of conceptual mod...
CSI4141 - Real Time Systems Design (3 un...

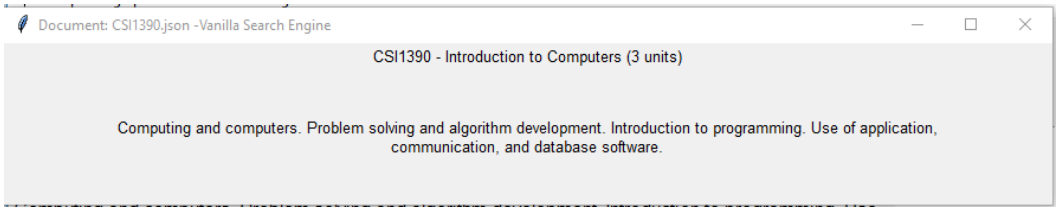

operating system-Vanilla Search Engine
Query: operating system
CSI4141 - Real Time Systems Design (3 un...
Definition of real-time systems; examples. Characteristics of real-time systems. Analysis framewo...
CSI5122 - Software Usability (3 units)...
Design principles and metrics for usability. Qualitative and quantitative methods for the evaluation...
CSI5134 - Fault Tolerance (3 units)...
Hardware and software techniques for fault tolerance. Topics include modeling and evaluation techniq...
CSI5311 - Distributed Databases and Tran...
Principles involved in the design and implementation of distributed databases and distributed transa...
CSI5314 - Object-Oriented Software Devel...
Issues in modeling and verifying quality and variability in object-oriented systems. Testable models...


Document: CSI5314.json -Vanilla Search Engine
CSI5314 - Object-Oriented Software Development (3 units)
Issues in modeling and verifying quality and variability in object-oriented systems. Testable models in model-driven and test-driven approaches. System family engineering. Functional conformance: scenario modeling and verification, design by contract. Conformance to non-functional requirements: goals, forces and tradeoffs, metrics. This course is equivalent to COMP 5104 at Carleton University.

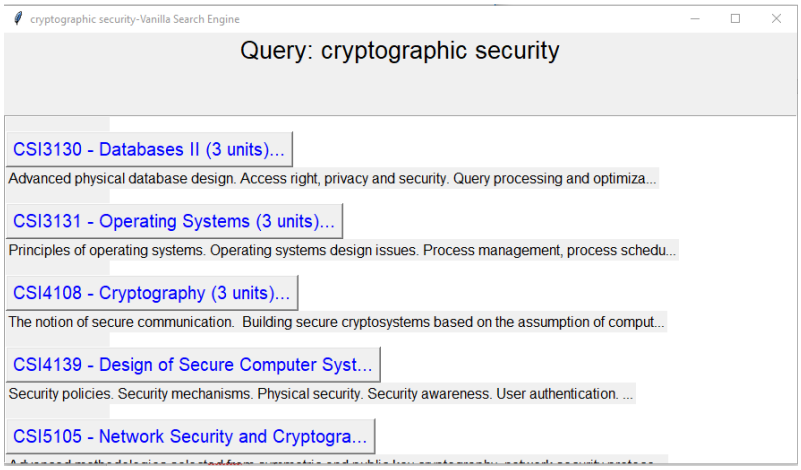
Query: computers graphical





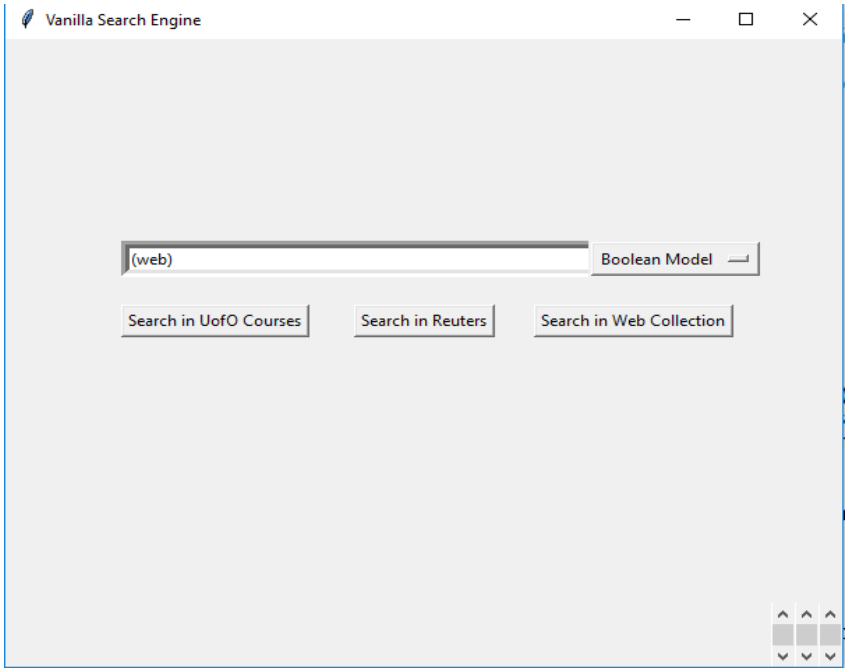


Query: cryptographic security



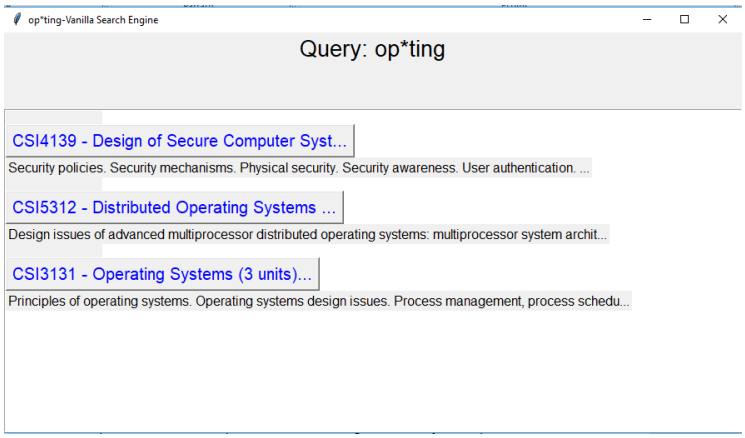
Additional queries

Query: (web)



Note with this query that the system can receive queries with parenthesis.

Query: op*ting



We can see that wildcards implementation with the system works fine.

REFERENCES

https://www.analyticsvidhya.com/blog/2017/03/read-commonly-used-formats-using-python/
https://stackoverflow.com/questions/2521482/getting-beautifulsoup-to-find-a-specific-p
https://www.tutorialspoint.com/python/python_reading_html_pages.htm
https://stackoverflow.com/questions/30147223/beautiful-soup-findall-multiple-class-using-one-query
https://stackoverflow.com/questions/13949637/how-to-update-json-file-with-python
https://github.com/CaptainSame/Search-Engine-in-Python/blob/master/gui.py
https://stackoverflow.com/questions/18337407/saving-utf-8-texts-in-json-dumps-as-utf8-not-as-u-escape-sequence
https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
https://stackoverflow.com/questions/35807433/import-nltk-no-module-nltk-corpus (you will need to install this module to run the code)
http://interactivepython.org/runestone/static/pythonds/BasicDS/InfixPrefixandPostfixExpressions.html
https://www.saltycrane.com/blog/2007/09/how-to-sort-python-dictionary-by-keys/
https://pypi.org/project/pythonds/ (You will need to install this module to run the code)
https://www.tutorialspoint.com/python/python_sets.html
http://carrefax.com/new-blog/2017/5/20/stackoverflow-how-can-i-generate-bigrams-for-words-using-nltk-python-library
https://pythonspot.com/python-set/